PATENT ABSTRACTS OF JAPAN

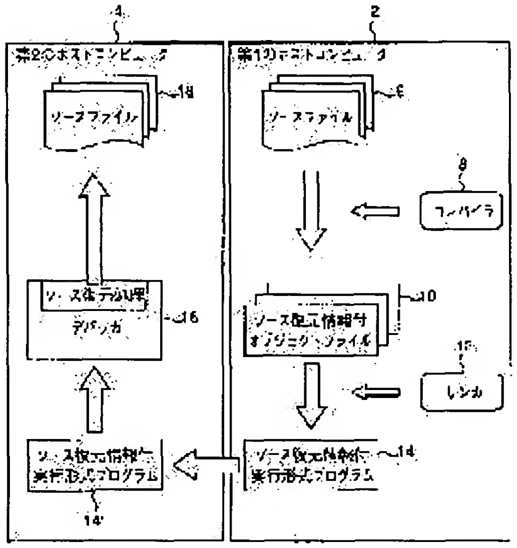
(11)Publication number : 2002-259121
(43)Date of publication of application : 13.09.2002

(51)Int.Cl. G06F 9/44
G06F 11/36
G06F 9/45
G06F 11/28

(21)Application number : 2001-054229
(22)Date of filing : 28.02.2001
(71)Applicant : RICOH CO LTD
(72)Inventor : KISHIKAWA JUN

(54) SOURCE LINE DEBAGGING DEVICE

(57)Abstract:
PROBLEM TO BE SOLVED: To restore a source program from an execute form program by a computer in which a debagger is operated.
SOLUTION: At the time of generating the execute form program, the information of defined functions and variables, the information of referred functions and variables, a start address, and an end address are included in each source file and argument information and the start address and end address of each function are included in each function. A source code is restored by utilizing the information for reverse compiling. A conversion correspondence table between the original source program an the restored source program in each row number is prepared and a source level can be debagged by a machine driven by a debagger.



LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

CLAIMS

[Claim(s)]
[Claim 1] The program development system which adds the information searched for by restoration of the source program by reverse compile in the above-mentioned object code and an execute-form program in the program development system which carries out compile processing of the source program file, generates object code, carries out joint processing of this object code further, and generates an execute-form program.
[Claim 2] the information required in restoration of the source program by the above-mentioned reverse compile - for every source file -- the program development system according to claim 1 which contains the starting address of - argument information for every further above-mentioned function, and - function, and the ending address of - function including the information on the information on the function defined and a variable, the function currently referred to, and a variable, - starting address, and - ending address
[Claim 3] The program development system according to claim 1 or 2 which makes each code of the above-mentioned execute-form program equipped with the line number of a source program.
[Claim 4] The program debugging system which restores a source program using the information required in the restoration of the source program by reverse compile added to this execute-form program based on the execute-form program generated by the claim 1 or the claim 3.
[Claim 5] The program debugging system according to claim 4 by which the correspondence relation between the line number with which the execute-form program generated by the claim 3 is equipped, and the line number in the source program generated after restoration is generated as a translation table.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] It is related with the system which performs source code debugging of the program performed on a microcomputer system, and the method of performing source code debugging.
[0002]
[Description of the Prior Art] In case the program which constitutes the system which works on a microcomputer, and these systems is developed, development is usually performed using a workstation or a personal computer (personal computer). At the time of these development, coding of the program is carried out by various high-class programming language, for example, C, C++, etc.
[0003] First, in the case of coding, in a personal computer etc., various text editors are used and a program is built by text data. These programs built are written in a file. Thus, generally the file holding the program as a text is called source file, and the program as a text is called source program. If the source program which is the content of a source file remains as it is so that it may be known well, it cannot be made to operate by computer.
[0004] Next, in order to operate these source files (source program) by computer, predetermined transform processing, i.e., compile processing, and predetermined link processing are needed. First, a source file is changed into the object file expressed by the machine language by compile processing. Furthermore, it is required to make it change into form, i.e., the form matched with the address of the computer which is made to combine the object file which exists as two or more modules, and real-works, that these object files can be operated by computer which actually works. The file of the form created here is called execute-form file (or executable file).
[0005] The above procedure is shown in drawing 5 . These things are attributes common not only to a program but various computer programs which work with a microcomputer, and are matters well-known to this contractor, of course.
[0006] By the way, when developing the program in connection with the system which operates with a microcomputer (microprocessor), as mentioned above, a source file is usually first created by the personal computer or workstation, then, similarly compile processing and link processing are further performed in a personal computer or a workstation.
[0007] Then, in order to mount an execute-form program in the object processor which is a microprocessor, usually the COP for an object processor downloads these execute-form program.
[0008] Here, the COP is equipped with a debugger, and debugging work will be done, carrying out test operation of the object processor. In this debugger, when it is going to perform source code (source program) debugging, though natural, the source program must exist around the COP (in namely, position which can be referred to). However, the machine (computer) for developing - source program and the COP for an object processor With the COP for the processor for - which may be another machine, completely The machine for developing - which may not hold the source program, in addition a source program, in order to save an unnecessary storage region (computer), The present condition is being unable to refer to a source program in many cases from the above-mentioned debugger, since [these] it says [not having any / the COP for an object processor / means of communications in many cases, and].
[0009] In JP,7-104978,A, by adding to an execute-form program, after compressing a source program, as long as an execute-form program exists, the structure which can restore a source program is built, and therefore, solution of the above-mentioned trouble is suggested. However, since the execute-form program is supporting the compression source program of the specified quantity, the trouble that effective use of a storage region cannot be aimed at remains.
[0010]
[Problem(s) to be Solved by the Invention] this invention aims at an execute-form program restoring a source program (file), and enabling it to perform source line debugging further in the host computer which does not have means of communications with the host computer (grade) which does not store the source program although stored, and stores the source program code. And it aims at realizing the above-mentioned thing, without changing most required capacity of an execute-form (program) file (increase).
[0011]
[Means for Solving the Problem] this invention is made in order to attain the above-mentioned purpose. The program development system according to claim 1 concerning this invention is a program development system which carries out compile processing of the source program file, generates object code, carries out joint processing of this object code further, and generates an execute-form program. In the program development system, the information searched for by restoration of the source program by reverse compile in the above-mentioned object code and an execute-form program is added.
[0012] the information required in restoration of the source program by reverse compile of the above [the program development system according to claim 2 concerning this invention] - for every source file -- it is the program development system according to claim 1 which contains the starting address of - argument information for every further above-mentioned function, and - function, and the ending address of - function including the information on the information on the function defined and a variable, the function currently referred to, and a variable, - starting address, and - ending address

[0013] The program development system according to claim 3 concerning this invention is a program development system according to claim 1 or 2 which makes each code of the above-mentioned execute-form program equipped with the line number of a source program.
[0014] The program debugging system according to claim 4 concerning this invention is a program debugging system which restores a source program using the information required in the restoration of the source program by reverse compile added to this execute-form program based on the execute-form program generated by the claim 1 or the claim 3.
[0015] The program debugging system according to claim 5 concerning this invention is a program debugging system according to claim 4 by which the correspondence relation between the line number with which the execute-form program generated by the claim 3 is equipped, and the line number in the source program generated after restoration is generated as a translation table.
[0016]
[Embodiments of the Invention] The gestalt of the suitable operation concerning this invention is explained referring to a drawing below.
[0017] Drawing 1 shows the sequence of the example of source code program debugging implementation processing concerning this invention. In drawing 1 , in the 1st host computer 2, a source program is created and it debugs in the 2nd host computer 4.
[0018] With the 1st host computer 2, one or more source programs 6 are created by the suitable text editor. Compile processing is carried out by the compiler 8, and each of the created source program 6 is changed into an object file 10. The information for restoration to a source program (henceforth source code restoration information) is added to each of the above-mentioned object file 10. That is, it is designed and the above-mentioned compiler 8 in this invention is formed so that source code restoration information may also be generated and added to object file 10 generate time.
[0019] Furthermore, with the 1st host computer 2, a linker 12 succeeds in joint processing of the above-mentioned object file 10, and the execute-form file 14 is formed. Source code restoration information is added also to this execute-form file 14. That is, the above-mentioned linker 12 in this invention also arranges and combines with execute-form file 14 generate time the source code restoration information added to an object file 10.
[0020] The above-mentioned execute-form file 14 is transmitted to the 2nd host computer 4, and is stored in the Records Department (not shown). When performing debugging processing in the 2nd host computer 4, the debugger 16 stored in the 2nd host computer 4 at the time of load processing of execute-form file 14' restores the source code program 18 per a function unit or source file. Source code program restoration is explained later.
[0021] It sets to the 2nd host computer 4, and it is only that the source code program 18 cannot be referred to. By using the debugger 16 stored in the 2nd host computer 4, source code (source program) debugging processing can be performed. That is, source code debugging processing which can be carried out in the 1st host computer 2, and same debugging processing may be carried out. Debugging processing is explained later.
[0022] Drawing 2 shows the example of the composition of the object file 10 (or execute-form file 14) to which source restoration information was added.
[0023] In the above-mentioned object file 10, ""source file information" 20 [required in order to restore a source program], and "function information" 22" are contained in addition to the source code debugging information (however, not shown) equipped from the former. Here, "source code debugging information" is information for being used with the debugger of the conventional technology and supporting debugging processing. Furthermore, although header information 24, the global symbol information 26, the local symbol information 28, and a program code (assembly code) 30 are contained in an object file 10 as shown in drawing 2 , it is the matter of common knowledge to this contractor that these are contained also in the conventional technology.
[0024] In the gestalt of suitable operation of this invention, each instruction code which constitutes the above-mentioned program code (assembly code) 30 has data of the line number of the original source program subordinately. For example, suppose that (1) is a part of original source program, and (2) is the assembly code in drawing 4 . It is that (1) will accompany the number of this "30" in each code of (2) supposing a line number is "30" in the original source program.
[0025] "source file information" 20 of drawing 2 -- every source file 6 - it is constituted by the information on the information on the function defined and a variable, the function currently referred to, and a variable, - starting address, and - ending address
[0026] "Function information" 22 of drawing 2 are constituted more by the starting address of - argument information and - function, and the ending address of - function.
[0027] Joint processing is performed by the linker 12 and, as for an object file 10, the execute-form program 14 is formed. This execute-form program 14 is also equipped with the composition (example) of drawing 2 , and is formed. however, the source file 6 of the program combined with "the source file information 20" with the composition of the execute-form program 14 after link processing although only the information about the source file 6 was included in "the source file information 20" in the composition of the object file 10 after compile processing -- the information about all is included
[0028] In the case of combination by the above-mentioned linker 12, conversion to unsolved address decision of a symbol and the absolute information on relocatable information is performed.
[0029] With the 2nd host computer 4, above-mentioned execute-form program 14' is loaded through a debugger 16. Restoration of a global variable, each function entry, and a return portion is first performed every source file 18 based on the source file information 20 and the function information 22 by the debugger 16. Drawing 3 is the example of the C source file which restored the time.
[0030] Then, the real program code from the entry position about each function to an end position (address) is extracted from "program code" 30 of execute-form file 14' using the data concerning the address currently recorded on the source file information 20 and the function information 22. Disassemble is performed from the extracted real program code (assembly code) 30.
[0031] In disassemble [above], suitably, the operand equivalent to the label of an assembly code is carried out based on variable information or argument information, and is changed into a symbol name. After changing into a symbol, in an assembly code, the line number of the source program of the origin which each assembly code accompanies uses the same thing as 1 lump, and carries out reverse compile of an assembly code group.

[0032] If the above-mentioned reverse compile (to source program) is completed about one function, it will restore about a consecutive function. If reverse compile, i.e., restoration processing, is completed about all functions, these functions will be adjusted every source file 18, and the whole source program will be restored.
[0033] Each of the assembly code 30 is accompanied by the line number of the original source program 6. The debugger 16 of the 2nd host computer 4 matches the line number in the source program 18 after restoration of the assembly code 30, and the line number of the above-mentioned original source program 6, and creates the translation table 32 of a line number.
[0034] For example, the assembly code of two lines of drawing 4 (3) stored in the Records Department (not shown) of the 2nd host computer 4 presupposes that all are accompanied by "30" as a line number of the original source program 6. Furthermore, the assembly code of these two lines is restored like drawing 4 (4), and after restoration presupposes that the line number was set to "40" and "41." At this time, "40" and "41" will be matched with a line number "30", and a translation table 32 like drawing 4 (5) will be generated.
[0035] As mentioned above, an object file 10 and the execute-form file 14, and 14' include "source line debugging information." Since this "source line debugging information" is generated according to the line number of the original source program 6, if it remains as it is, it cannot respond to the source program 18 after restoration. Here, if the above-mentioned translation table (32) is used, "source line debugging information" and the source program 18 after restoration can be matched, and source line debugging can be performed in the source program 18 after restoration using "source line debugging information."
[0036]
[Effect of the Invention] By using this invention, the following thing becomes possible. That is, in the host computer which does not have means of communications with the host computer (grade) which does not store the source program although stored, and stores the source program code, an execute-form program can restore a source program (file), and can perform source line debugging further. And the above-mentioned thing becomes possible, without changing most required capacity of an execute-form (program) file (increase).

[Translation done.]

DESCRIPTION OF DRAWINGS

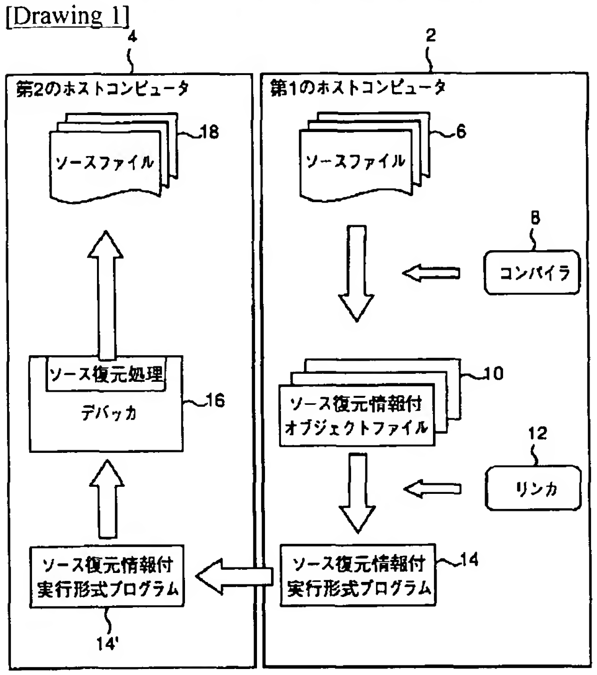
[Brief Description of the Drawings]
[Drawing 1] The sequence of the example of source code program debugging implementation processing concerning this invention is shown.
[Drawing 2] The example of the composition of the object file to which source restoration information was added, or an execute-form file is shown.
[Drawing 3] It is the example of the C source file to which restoration to a global variable, and each function entry and return portion was carried out.
[Drawing 4] They are the example of the original source program and the source program restored [which were restored and was execute-form-programmed], and the example of a translation table.
[Drawing 5] It is the general procedure which creates an execute-form file from a source program.
[Description of Notations]
2 [... A compiler, 10 / ... An object file with source restoration information (object program),, 12 / ... A linker, 14, 14' / ... An execute-form file with source restoration information (execute-form program),, 16 / ... A debugger, 18 / ... An after / restoration / source file (source program),, 20 / ... Source file information, 22 / ... Function information, 30 / ... An assembly code (program code), 32] ... The 1st host computer, 4 ... The 2nd host computer,

---

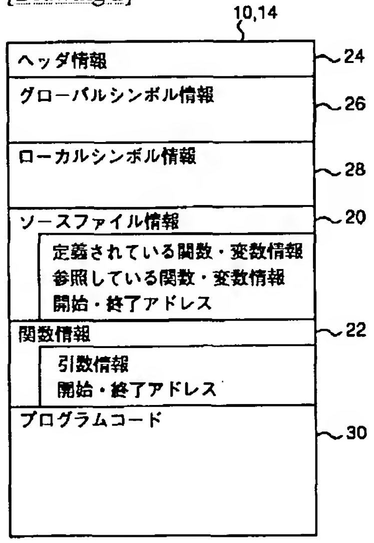
[Translation done.]

DRAWINGS

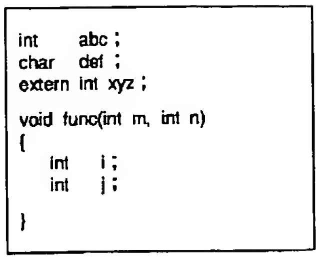
[Drawing 1]



[Drawing 2]



[Drawing 3]



```
int     abc ;
char    def ;
extern int xyz ;

void func(int m, int n)
{
    int   i ;
    int   j ;

}
```

[Drawing 4]

(1)

```
30      A=B ; C=D ;
```

(2)

```
        MOV    A , B            —— (30)

        MOV    C , D            —— (30)

           ⇩  転送

(3)
        MOV    A , B               (30)

        MOV    C , D               (30)
```

(4)

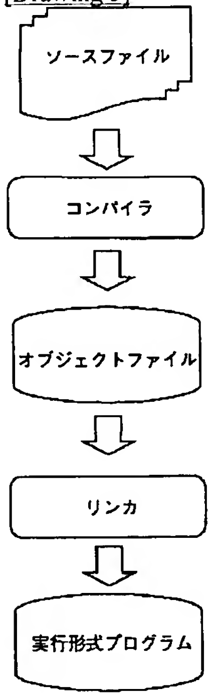
```
40      A=B ;

41      C=D ;
```

(5) 32

30	40
30	41

[Drawing 5]



[Translation done.]